2015 IEEE 27th International Conference on Tools with Artificial Intelligence

A Multi-Population Genetic Algorithm for UAV Path Re-Planning under Critical Situation

Jesimar da Silva Arantes,
Márcio da Silva Arantes,
Claudio Fabiano Motta Toledo
University of São Paulo, USP
São Carlos, Brazil
jesimar.arantes@usp.br,
{*marcio, claudio*}*@icmc.usp.br*

Brian Charles Williams
Massachusetts Institute of Technology, MIT
Cambridge, USA
williams@mit.edu

***Abstract*—This paper studies the path planning for Unmanned Aerial Vehicles (UAVs) under critical situations, where the aircraft has to execute a hard landing. Such critical situations can be provoked by equipment failures or extreme environmental situations that demand the UAV to abort the mission running and to land the aircraft without risk for people, properties and itself. First, a mathematical formulation is introduced to describe this problem. A planner system is proposed based on a multi-population genetic algorithm and a greedy heuristic. Computational results are conducted over a large set of scenarios with different levels of difficulty. Also, some simulations are executed using FlightGear simulator to illustrate the UAV's behaviour when landing under different wind velocities. The results achieved indicate the greedy heuristic is able to define faster feasible landing paths, whose quality can be improved by the evolutionary approach always within a short computation time.**

***Keywords*-Evolutionary Computing; Decision Optimization; Unmanned Aerial Vehicles; Path Planning; Uncertainty**

I. Introduction

The path planning for Unmanned Aerial Vehicles (UAVs) under the occurrence of critical situations is studied in this paper, where a greedy heuristic and a genetic algorithm are applied to define emergency routes for landing. In this context, the mission running on the aircraft needs to be aborted and the current route has to be re-planned by the proposed methods. The main objective becomes to land the UAV without risk to the safety of people, properties and aircraft itself. Critical situations can be related to failures in the equipments of the UAV or extreme environmental situations. In the case of equipment failures, it is possible to have technical problems with sensors, motor crash, battery overheating, among others. Examples of extreme environmental situations can be strong turbulence, unforeseen storm approximation and detection of another aircraft. The mission can be aborted leading the UAV to return base or even to execute a hard landing. This paper will evaluate some critical situations caused by equipment failures that demand a hard landing. The use of UAVs increased in the last years and discussions have begun about their commercial application and risks related to their operation. The probability of failure for this type of aircraft should be less than or equal the currently accepted in general aviation [4]. The authors in [7] argue that safety is the most important factor to ensure the integration of UAVs to airspace, so ways to bring back on board relevant features lost with the absence of pilots have to be considered. This is the case when a decision about path re-planning has to be made under critical situations, where a viable route must be defined within a short time.

Genetic algorithms, as well as other heuristic and evolutionary techniques, have been successfully applied in real world problems where near-optimal solutions become more relevant than time-consuming optimal solutions [10]. Thus, the main contribution of this paper is to describe the problem of path re-planning under an emergency landing and to propose fast approaches to solve it. A mathematical formulation is introduced to describe this problem and a multi-population genetic algorithm (MPGA) is applied as path planner. It is also introduced a greedy heuristic (GH), whose objective is to build fast feasible paths. The performance of GH is evaluated by itself and as an initialization operator for MPGA. The paper is organized as follows: section 2 describes some related work and section 3 defines the problem to be studied. The proposed methods are presented in section 4 and computational results achieved are reported in section 5. The conclusions of this work are in section 6.

II. Related Work

The mission planning for autonomous vehicles is studied in [6], [2], [12]. A system able to plan a sequence of discrete actions and continuous controls for air and underwater autonomous vehicles is presented in [6]. The authors in [2] evaluate risks during the path planning for UAV, where the risk of collision with obstacles must be within a certain margin of safety. The mission planning is studied in [12], where a new planner system is proposed that introduces improvement in the treatment of risk allocation and incorporates scheduling for the tasks to be executed.

1082-3409/15 $31.00 © 2015 IEEE
DOI 10.1109/ICTAI.2015.78

The path re-planning during a critical situation is studied in [9], where a system helps aircraft pilots to determine the best landing site for a damaged aircraft. After the critical failure of the aircraft, the pilot must first recover control of the aircraft, so the goal becomes to find the best place for the emergency landing. The Emergency Landing Planner (ELP) system proposes routes as well as possible landing sites for the aircraft using an A* hybrid algorithm. In a later work [8], the behavior of the A* hybrid algorithm proposed in [9] was analyzed through its application in a real scenario. The authors conducted tests in a flight simulator for large aircraft, where the A* hybrid algorithm was used to determine the landing site. The simulation considers problems occurring between 1 to 3 minutes of flight, where the ELP algorithm is triggered to assist pilots.

Evolutionary algorithms have been used in path planning for UAVs. The author in [13] applies genetic algorithms with Voronoi diagram for path planning of UAVs. It is given emphasis in the new mutation strategy proposed that was separated in global and local random diversity. The Voronoi diagram is used in the initial phase when the population is created. Metrics to compare path planners for UAVs are proposed by [1], where it is taken into account the complexity and peculiarities of the problem handled. The authors evaluate the performance of three techniques: Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Differential Evolution (DE). The results reported that GA is the best method, followed by PSO and DE. Differential Evolution is applied in [15] to path planning for UAV in three-dimensional environments. The UAV has to avoid no-fly zones, radars zones, missiles and anti-aircraft guns. This paper proposes the use of MPGA, where individuals are hierarchically structured in trees. This approach was introduced by [5] with individuals following a hierarchy based on their fitness value in a ternary tree. Such approach has been applied to solve problems in different contexts like glass container production scheduling [14], ordering microarray data [11], among others, with relevant results reported.

The studies conducted in [6], [2], [12] are advances in the treatment of risk for planning and ensure that the planned mission is executed within a safety margin, considering the inaccuracies of the equipment and the environment. However, path re-planning under critical situations is not considered as proposed in the present paper. Therefore, the present paper introduces a mathematical formulation for this path re-planning problem based on that described in [2] for path planning with risk allocation. When considering failure in aircraft, the current study is closer to that described in [9], but our study is dealing with UAVs and it has also a different set of associated failures, which demand an urgent aircraft landing. The aircraft in [9] is still able to flight even during failure.

III. Problem Description

The problem is described in this section from the situation illustrated by Figure 1. There is a scenario composed by two populated regions (houses), woods, a plain area, an area under storm, an airport, a scenic region and the UAVs runway. Suppose a mission to be performed by the UAV followed by a critical situation as described next:

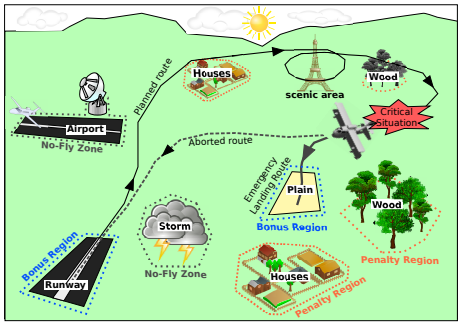

Figure 1: Illustrative scenario for mission planning.

Mission: Starting from the UAV runway, it must reach the scenic region and remain in it taking pictures for a while. Next, the aircraft must return to the runway. In all the way, it should stay in a safety region, avoiding no-fly zones as storm and airport. The aircraft can fly over populated and wood areas.

Critical situation: During the mission execution, the UAV systems detect a problem like battery overheating. Such systems are able to abort the mission and trigger the algorithm to re-plan the current route. Such re-planning will from now seek to minimize the likelihood of damage during landing, considering information from the mentioned regions and the limitations caused by the aircraft's problem.

A. Sets of Regions

Several regions can be mapped before the mission execution and classified in sets of regions, based on their characteristics. These sets are separated according to the probability of landing the aircraft in one of their regions. A total of four sets are defined for the types of regions considered by this work.

1) **No-Fly Set (ϕ_n):** The aircraft cannot fly over and land in the regions of this set. Regions within this set can be represented by airports, military base and other areas with restrictions on the UAV flight.
2) **Navigable with Penalty Set (ϕ_p):** The UAV can fly over regions on this set, but it is not desired landing on them. The regions of this set may represent populated regions, factories, forests, among others.
3) **Navigable and Bonus Set (ϕ_b):** The UAV can fly and it is desired landing on regions of this set. This set contains flat and suitable regions for landing as grassy areas or fields with grounder plantations.

4) **Remainder Set (ϕ_r):** The aircraft can fly and land in these regions. This set represents the remain areas that are not classified for landing. There are no restrictions for flight or landing in these regions.

B. Critical Situations

The re-planning algorithm must also be informed about the kind of limitation imposed on the aircraft by the detected problem. In this work, some critical situations to aircraft operation were considered:

1) **Motor Failure (ψ_m):** The motor m presents problems and stops functioning. In this case, a suitable region for the landing has to be found while the aircraft is hovering.
2) **Battery Failure (ψ_b):** The battery b has an overheating, where all the controls work, but the UAV should land as soon as possible.
3) **Aerodynamic Surfaces Failure type 1 (ψ_{s^1}):** The aircraft fails in one of the wings s^1 that makes it able only to turn left.
4) **Aerodynamic Surfaces Failure type 2 (ψ_{s^2}):** The aircraft fails in one of the wings s^2 that makes it able only to turn right.

C. Problem Modeling

We present a mathematical formulation to summarize all aspects of the problem addressed. This formulation comes from a similar modeling described in [2] for path planning with risk allocation.

Parameters:

- $\phi_j = \{Z^1_{\phi_j}, Z^2_{\phi_j}..., Z^{|\phi_j|}_{\phi_j}\}$: Set of regions with $j \in \{n, p, b, r\}$;
- $Z^i_{\phi_j}$: ith region of the set ϕ_j;
- $|\phi_j|$: Number of regions in set ϕ_j;
- C_{ϕ_j}: Cost of landing in set ϕ_j;
- T: Number of time steps to land the UAV;
- Δ: Probability of UAV violate a region in the set ϕ_n;
- F_Ψ: State transition function for a given type of failure Ψ_k with $k \in \{m, b, s^1, s^2\}$;
- ΔT: Time discretization established in the simulation;
- ω_t: State-independent disturbance at time step t.

Decision variables:

- x_t: Set of states of the UAV ($x_t = [p^x_t, p^y_t, v_t, \alpha_t]^T$);
- p^x_t: Position in the x-axis of the UAV at time step t;
- p^y_t: Position in the y-axis of the UAV at time step t;
- v_t: Velocity at time step t of the aircraft;
- α_t: Angle of the UAV in time step t from x-axis;
- u_t: Set of UAV controls ($u_t = [a_t, \varepsilon_t]^T$);
- a_t: Acceleration of the UAV at time step t;
- ε_t: Angular variation of the UAV.

$$Minimize \sum_{i=1}^{|\phi_p|} (C_{\phi_p} \cdot P(x_T \in Z^i_{\phi_p})) - \sum_{i=1}^{|\phi_b|} (C_{\phi_b} \cdot P(x_T \in Z^i_{\phi_b})) \quad (1)$$

subject to:

$$x_{t+1} = F_\Psi(x_t, u_t) + \omega_t \qquad \forall\, t = 0, 1, \dots T \quad (2)$$

$$P\left(\bigwedge_{t=0}^{T} \bigwedge_{i=1}^{|\phi_n|} x_t \notin Z^i_{\phi_n} \right) \geq 1 - \Delta \quad (3)$$

A discrete and finite series of time steps $t = 0, 1, ..., T$ is assumed to land the aircraft. The decision variable x_t represents the state of UAV at time t, which is given by its coordinate at Cartesian plane (p^x_t, p^y_t), velocity (v_t) and angular direction on this plane (α_t). The variable u_t has the controls at t, where the acceleration (a_t) and angular variation (ε_t) applied over the aircraft are defined.

The objective function (1) aims to minimize damage executing a safety landing, so it has two parts: penalty and bonus. The first part has the penalties (C_{ϕ_p}) given for a route that lands the aircraft at the last time step ($t = T$) within penalty set regions ($x_T \in Z^i_{\phi_p}$). The second part has the rewards (C_{ϕ_b}) given for a route that lands the aircraft within safety regions ($x_T \in Z^i_{\phi_b}$). The function $P(...)$ used in this formulation represents the probability of the aircraft in the state x_t belongs or not to a particular region in ϕ_i.

Constraints (2) describes the transition states of the UAV, where the dynamic of the UAV states at time $t + 1$ are defined from positions and controls applied at time t added by uncertainty ω_t from external factors. The function F_Ψ can depend on each type of critical situation addressed (Ψ). As proposed in [2], it is assumed that the uncertainty ω_t follows a Gaussian white noise distribution $\omega_t \sim \mathcal{N}(0, Q)$ with covariance matrix Q. It is also assumed a Gaussian distribution for the initial position with mean $\hat{x}_0$ and covariance matrix Σ_{x_0}, so $x_0 \sim \mathcal{N}(\hat{x}_0, \Sigma_{x_0})$. Moreover, the future states follow a Gaussian distribution and x_t becomes a random variable.

Thus, the location of the UAV is not precise and there is always the risk of it deviates from its route and reaches a non-fly area, but a limit for such risk can be considered during the path planning. Constraints (3) describe the probability ($1 - \Delta$) of the UAV being out of regions that belong to set ϕ_n. The final trajectory will be returned by set of waypoints for the landing route.

IV. Methods

This section presents proposed methods. First, the representation of solution (encoding) adopted as well as fitness function and operators that deal with such representation are explained. Next, the methods are described from their pseudocodes.

A. Codification, Operators and Fitness

The solution of the problem is encoded as real values for the controls $u_t = [a_t, \varepsilon_t]^{'}$ applied to the aircraft at each instant t. Four initialization operators are defined to create such codification: random, short curves, short acceleration and greedy. The random initialization generates values with

uniform distribution for $a_t \in U[a_{min}; a_{max}]$ and $\varepsilon_t \in U[\varepsilon_{min}; \varepsilon_{max}]$. The short curves operator takes values for $a_t \in U[a_{min}; a_{max}]$, but it draws reduced values for angular variation with $\varepsilon_t \in U[0.25\varepsilon_{min}; 0.25\varepsilon_{max}]$. The short acceleration does the opposite, i.e., it takes $\varepsilon_t \in U[\varepsilon_{min}; \varepsilon_{max}]$ and $a_t \in U[0.25a_{min}\ ; 0.25a_{max}]$. These two operators give more chance to apply smoother controls for changes over the direction and acceleration of the aircraft. The greedy initialization applies the greedy heuristic, explained in the next section, to generate a set of controls that already guarantees to reach bonus regions.

The crossover operators are average, geometric, arithmetic, OX and BLX-α. The mutation operators are uniform, limit and creep. These are classical real-coded operators from literature [10]. Previous computational experiments evaluating these operators were conducted. From these results, the better strategy is to apply all operators, which means always to select randomly one among the five crossovers to be applied every time a new individual is created. Also, one between the three mutation operators is randomly chosen every time the mutation rate is satisfied.

The transition function $F_\Psi(x_t, u_t)$ is used to decode the controls using Equation 4. This function approximates the motion of a fixed-wing UAV in a 2-D space, wheres F_t^d and m indicate resistance of the air and the mass of the UAV, respectively.

$$F_\Psi(x_t, u_t) = \begin{bmatrix} p_t^x + v_t \cdot cos(\alpha_t) \cdot \Delta T + a_t \cdot cos(\alpha_t) \cdot (\Delta T)^2/2 \\ p_t^y + v_t \cdot sin(\alpha_t) \cdot \Delta T + a_t \cdot sin(\alpha_t) \cdot (\Delta T)^2/2 \\ v_t + a_t \cdot \Delta T - \frac{F_t^d}{m} \cdot \Delta T \\ \alpha_t + \varepsilon_t \cdot \Delta T \end{bmatrix} \quad (4)$$

Once x_t is a random variable ($x_t \sim \mathcal{N}(\overline{x}_t, \Sigma_t)$), the authors in [2] show that the next expected state ($\overline{x}_{t+1}$) depends only of the current expected state ($\overline{x}_t$) and nominal controls applied ($\overline{u}_t$), so we can use the transition function to calculate all the next expected states ($\overline{x}_{t+1} = F_\Psi(\overline{x}_t, \overline{u}_t)$). However, the uncertainty Σ_t around the expected state $\overline{x}_t$ grows every time step. The authors in [12] introduced a close-loop control approach to define control inputs from a nominal control input $\overline{u}_t$. It is applied a correction that slows down the growth of the uncertainty Σ_t. In this work, we assume that the uncertainty Σ_t at any time t is constant ($\Sigma_t = Q$). If it is known that the uncertainty about the state x_t at any time step t is given by $x_t \sim \mathcal{N}(\overline{x}_t, \Sigma_t)$, it is possible to calculate the probability function $P(x_t \in Z_\phi^i)$ using a lookup table for a Gaussian distribution. The covariance matrix used in this work is given by Equation 5 where $\sigma = 10$ meters is assumed.

$$\Sigma_t = Q = \begin{bmatrix} \sigma^2 & 0 & 0 & 0 \\ 0 & \sigma^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (5)$$

The aircraft failures are described from changes in Equation 4. For critical situation ψ_m, the acceleration is set null ($a_t = 0$) in F_{Ψ_m} once it is not possible to accelerate the UAV during a motor failure. The angular variation becomes only positive ($\varepsilon_t \in [0, \varepsilon_{max}]$) for ψ_{s^1}, when the UAV is able only to turn left. On the other hand, $\varepsilon_t \in [\varepsilon_{min}; 0]$ when the UAV is able only to turn right (ψ_{s^2}). There is no change in Equation 4 for critical situation ψ_b, once the UAV with a battery failure has only to quickly land, so the Equation 13 is added in the fitness function as explained latter.

The decoding procedure described will return the set with all waypoints to land the UAV by constraints (2) in section III-C. It is possible to land the aircraft without spending all time steps available, so we can land it in a time step K such that $0 < K \leq T$. This trajectory is a solution for this problem and it will be evaluated by a fitness function described by Equation 6.

$$\begin{aligned} fitness = f_{Landing_{\phi_b}} + f_{Landing_{\phi_p}} + f_{Flight_{\phi_n}} \\ + f_{Curves} + f_{DistUAV_{\phi_b}} + f_{Violated_T} + f_{\psi_b} \end{aligned} \quad (6)$$

Equations 7 and 8 define reward and penalization to land, respectively, in bonus and penalized regions.

$$f_{Landing_{\phi_b}} = -C_{\phi_b} \cdot \sum_{i=1}^{|\phi_b|} (P(x_K \in Z_{\phi_b}^i)) \quad (7)$$

$$f_{Landing_{\phi_p}} = C_{\phi_p} \cdot \sum_{i=1}^{|\phi_p|} (P(x_K \in Z_{\phi_p}^i)) \quad (8)$$

Equation 9 penalizes landing or flying over no-fly regions, while Equation 10 and 11 give more chance to paths that avoid unnecessary bends and with shortest distances to bonus regions, respectively.

$$f_{Flight_{\phi_n}} = C_{\phi_n} \cdot max(0, 1 - \Delta - P\left(\bigwedge_{t=0}^{K} \bigwedge_{i=1}^{|\phi_n|} x_t \notin Z_{\phi_n}^i\right)) \quad (9)$$

$$f_{Curves} = \frac{1}{|\varepsilon_{max}|} \cdot \sum_{t=0}^{K} \|u_t\| \cdot |\varepsilon_t| \quad (10)$$

$$f_{DistUAV_{\phi_b}} = shortestDist(\overline{x}_K, Z_{\phi_b}) \quad (11)$$

If the aircraft has final velocity greater than its minimum value, it is not landing in fact. Thus, Equation 12 avoids paths where the UAV did not land, besides it reaches the bonus region.

$$f_{Violated_T} = \begin{cases} C_{\phi_b} & , v_K - v_{min} > 0 \\ 0 & , \text{otherwise} \end{cases} \quad (12)$$

If there is a battery failure, Equation 13 is added to the fitness function. The aim is to reduce the number of waypoints instead of spending several time steps for aircraft landing.

$$f_\psi = \begin{cases} C_{\phi_b} \cdot 2^{\frac{(K-T)}{10}} & , \psi = \psi_b \\ 0 & , \text{otherwise} \end{cases} \quad (13)$$

B. GH and MPGA

A greedy heuristic (GH) was developed aiming to have a method simple enough to achieve a solution within a short time. Such approach can be worthwhile for landing under critical situations, where a fast decision-making is necessary and a viable solution becomes more interesting than time-consuming better solutions. Algorithm 1 describes the greedy heuristic.

Algorithm 1: Greedy Heuristic.

```
1 begin
2     RouteLanding route[] ← RouteLanding()[map.|φ_b|];
3     for i = 1 to map.|φ_b| do
4         initialize(route[i], map.Z^i_φb);
5         evaluate(route[i]);
6     RouteLanding bestRoute ← getBestRoute(route);
7     return bestRoute;
```

The basic operation of this heuristic is to generate a candidate solution for each one of the bonus regions ($map.|\phi_b|$). This candidate solution is determined rotating the aircraft until it reaches the same direction of a bonus region ($map.Z^i_{\phi_b}$). Next, it is calculated one trajectory straight to this region.

The rotation occurs to the side with lower angle between the UAV and a straight line to a bonus region. This is illustrated by angles $\lambda_1 < \lambda_1^{'}$ for region b_1 in Figure 2, where the rotation occurs towards the angle λ_1.

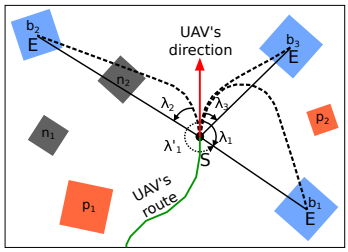

Figure 2: Candidate routes using Greedy Heuristic.

The best trajectory defined among all bonus regions is returned by the GH. The candidate solutions are evaluated according to their performance in Equation 6. In Figure 2, the best solution is that landing in region b_3, which is the closest region to the UAV. The solution that lands on b_1 has slightly worse performance and the solution that lands on b_2 violates the not navigability constraint, so it has low quality. As already mentioned, GH can be used as an initialization operator by MPGA. In this case, a bonus region is randomly selected by GH and a trajectory is built. The controls defined for this trajectory will be an individual in the MPGA.

Algorithm 2 describes the MPGA proposed in this work. The population is first initialized and all individuals are evaluated. Next, these individuals are structured in tree as illustrated by Figure 3. The hierarchy is represented by the position of the individuals (nodes) in the clusters of the tree, where the leader has better fitness than its followers. The best individual is represented by the root node and the worst individuals are the leaf nodes.

Algorithm 2: Multi-Population Genetic Algorithm.

```
1  begin
2      repeat
3          for i = 1 to numPop do
4              for j = 1 to numIndividuals do
5                  initialize(pop(i).ind(j));
6                  evaluate(pop(i).ind(j));
7              organize(pop(i));
8              repeat
9                  for j = 1 to rateCross × numIndividuals do
10                     select(parents);
11                     child ← crossover(parents);
12                     mutation(child);
13                     evaluate(child);
14                     add(child, pop(i));
15                 organize(pop(i));
16             until converge(pop(i));
17         for i = 1 to numPop do
18             migrate(pop(i));
19     until reach(stoppingCriterion);
20     RouteLanding bestRoute ← getBestRoute(pop);
21     return bestRoute;
```

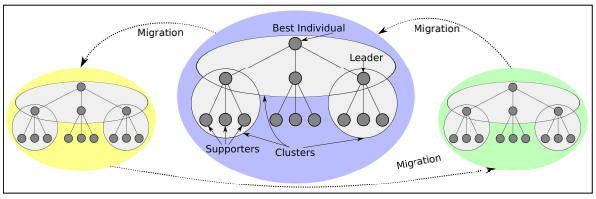

Figure 3: Hierarchical tree structure.

The evolution process selects randomly two individuals for reproduction, which are always a leader node and one of its followers, respectively. The new individual generated, if it has better fitness, replaces the worst parent. After the new individuals insertion, the population is hierarchically restructured where better individuals become leader in their clusters. The population converge when no individuals are inserted after $rateCross \times numIndividuals$ attempts. When all populations converge, a migration operator is executed and they are reinitialized, except by the best individual of each population. The migration operator sends a copy of the best individual from population i to population $i + 1$. The stop criterion is the number of fitness evaluations.

V. Computational Results

Maps with different scenarios were randomly generated to validate the landing path planners. For this purpose, an automatic map generator was developed, from one described in [2], based on the proportion of regions belonging to the

navigable and bonus set ϕ_b along with map density. It is considered a map with dimensions $1000m \times 1000m$ and the critical situation always happens in the center of the map at position $(p_0^x, p_0^y) = (0; 0)$. Three levels of difficulty are considered taking the proportion of regions in ϕ_b. The first level is the *Easy Maps* (M_E), where there are many bonus regions, a median level of penalty (ϕ_p) and few no-fly (ϕ_n) regions. The second level is named *Normal Map* (M_N) with a balanced number of regions belonging sets ϕ_n, ϕ_p and ϕ_b. The *Hard Map* (M_H) level has maps with many regions in set ϕ_n, a median amount in set ϕ_p and few regions in ϕ_b. The other two levels of difficulty take into account the density of the map. The level *Coverage 25%* ($C_{25\%}$) defines maps where the regions are more sparse, so they will occupy about 25% of the total area of the map. The level *Coverage 50%* ($C_{50\%}$) defines maps with regions that will occupy about 50% of the total area. For these criteria, it is possible to define six group of instances (I1-I6) from which 100 maps are generated for each instance as shown by Figure 4.

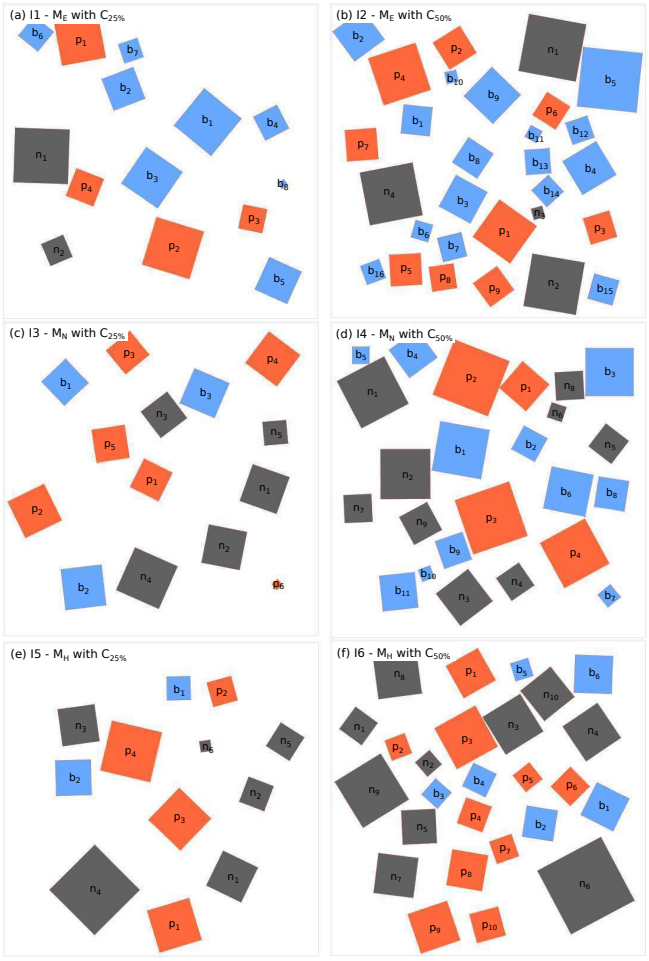

Figure 4: Automatically generated maps.

Table I shows parameters of the problem, where those for UAV are based on data of the UAV model Tiriba [3]. The experiments were performed with machines under Linux-Ubuntu 13.10, Intel Core i5 processor, 64 bits, 1.80 GHz and 4 GB RAM. Results of a previous parameter tuning for the MPGA are available at web[1]. From these results, MPGA is set with 3 populations, where each one has 13 individuals disposed in ternary trees in a total of 39 individuals. The evolutionary process applies 0.5 and 0.75 of crossover and mutation rates, respectively, and the stop criterion is 10,000 fitness evaluations.

Table I: Settings of the UAV and weights of penalties used in the experiments.

Parameters	Value	Parameters	Value
p_0^x, p_0^y (m)	$(0; 0)$	ΔT (s)	1
v_0 (m/s), α_0 $(^\circ)$	$24, 90$	Δ	0.001
v_{min}, v_{max} (m/s)	$[11.1;\ 30.5]$	C_{ϕ_b}	2000
$\varepsilon_{min}, \varepsilon_{max}$ $(^\circ/s)$	$[-3;\ 3]$	C_{ϕ_p}	8000
a_{min}, a_{max} (m/s^2)	$[0.0;\ 2.0]$	C_{ϕ_n}	100000
T (s)	60	C_{ϕ_r}	0

Two versions of MPGA are evaluated, one without applying GH as an initialize operator (MPGA1) and another applying it as operator (MPGA2). Table II shows the results following each type of critical situation associated with the UAV (Ψ), and the level of difficulty defined by Instances. The number of times the UAV landed in bonus (Landed ϕ_b) and penalty regions (Landed ϕ_p) are shown. It is also reported the number of infeasible paths, which happen if the UAV lands in remain regions (ϕ_r) or it can fly over a non-fly zone when landing in ϕ_b or ϕ_p. Let's give an example from the results reported for motor problem (ψ_m) in instances I1 (*Easy Maps* with *Coverage 25%*). In this case, GH is able to land the UAV for 79 out of 100 maps in bonus regions and 21 out of 100 maps landings in remainder regions, so the UAV does not land in a penalty region in this case. MPGA1 and MPGA2 return, respectively, 81 and 90 routes that land the UAV in a bonus area for I1.

MPGA2 outperforms all other approaches from the results reported in Table II. It landed the UAV in bonus area for 79.2% of the maps on average, when there is a motor failure, against 67.8% and 68.8% of GH and MPGA1, respectively. GH, MPGA1 and MPGA2 have their better performance dealing with battery overheating problem. MPGA2 landed the aircraft safely in 97.8% of maps, while GH returned 89.5% and MPGA1 88.8% . For flaws in both wings, MPGA2 is the only approach able to land the aircraft in bonus areas for more than 80% of maps. GH presents the worst performance for the majority of results. For all critical situations, the performance of MPGA1 is not much better than GH. This indicates that to evolve paths, without some previously defined route for bonus regions, is not a better strategy. All methods returned a large amount of infeasible solution when dealing with flaws in both wings.

[1]http://lcrserver.icmc.usp.br/projects/uav/wiki

Table II: Result obtained for the GH, MPGA1 and MPGA2 for all critical situations and all instances.

		GH			MPGA1			MPGA2		
Ψ	Inst.	ϕ_b	ϕ_r	Inf.	ϕ_b	ϕ_r	Inf.	ϕ_b	ϕ_r	Inf.
ψ_m	I1	79	21	0	81	19	0	90	10	0
	I2	92	6	2	92	7	1	96	3	1
	I3	58	39	3	60	39	1	71	28	1
	I4	86	12	2	84	16	0	96	4	0
	I5	30	52	18	36	64	0	40	60	0
	I6	62	28	10	60	33	7	82	15	3
	Avg	67.8	26.3	5.8	68.8	29.7	1.5	**79.2**	20.00	*0.83*
ψ_b	I1	99	0	1	100	0	0	100	0	0
	I2	97	0	3	99	0	1	99	0	1
	I3	93	3	4	94	5	1	99	0	1
	I4	98	0	2	99	0	1	100	0	0
	I5	67	5	28	73	27	0	94	6	0
	I6	83	0	17	68	17	15	95	2	3
	Avg	89.5	1.3	9.2	88.8	8.2	3.0	**97.8**	1.3	*0.8*
ψ_{s1}	I1	81	8	11	90	8	2	91	7	2
	I2	88	0	12	89	0	11	93	0	7
	I3	68	16	16	76	18	6	86	8	6
	I4	82	1	17	84	3	13	89	0	11
	I5	41	23	36	49	46	5	67	28	5
	I6	56	0	44	46	23	31	78	4	18
	Avg	69.3	8.0	22.7	72.3	16.3	11.3	**84.0**	7.8	*8.2*
ψ_{s2}	I1	90	4	6	94	4	2	99	0	1
	I2	90	0	10	95	1	4	95	1	4
	I3	70	20	10	79	16	5	92	5	3
	I4	87	1	12	83	8	9	94	0	6
	I5	40	17	43	62	35	3	74	24	2
	I6	61	3	36	57	13	30	76	4	20
	Avg	73.0	7.5	19.5	78.3	12.8	8.8	**88.3**	5.7	*6.0*
	-	74.9	10.8	14.3	77.1	16.7	6.2	**87.3**	8.7	*4.0*

The difference among the maps does not affect the processing time of the methods, since the stop criterion established was the limit of $10,000$ evaluations. GH is the fastest method as expected spending on average 0.07 sec. and no longer than 0.12 sec. in the worst case to return a solution. MPGA1 and MPGA2 spent on average around 1 sec. to conclude their execution and no longer than 1.5 sec. in the worst case. Figure 5 shows four routes determined by the planner MPGA2 in a map M_N with coverage $C_{25}\%$. In this scenario, the aircraft is flying towards north direction when a failure occurs. For each example, it is presented the original routes as well as the point S where the critical situation happens and the point E where the aircraft lands. The circles represent the uncertainty related with the UAV position as described in sections III and IV.

From one of the paths provided by MPGA2, a simulation is also executed aiming to illustrate the behaviour of the aircraft following such path under an environment with winds. The FlightGear (FG) was chosen once it is an open-source simulator with many resource available. FG was set to simulate the flight dynamic of Cessna 172, but an autopilot was coded to control automatically the Cessna 172 flight. Thus, the simulation is not conducted by a human pilot. Figure 6 shows the framework that integrates FG, autopilot and MPGA2 path. MGPA2 sends the path to the autopilot that aborts the current path and starts a new path. The autopilot executes the MPGA2 path based on the environment and aircraft simulations from FG. The green line is the original path executed by the aircraft, and the red path is the re-planning path for emergency land.

Two maps from Instances I3 (M_N with $C_{25\%}$) and I6 (M_H with $C_{50\%}$) were selected to execute the simulations where a battery failure ψ_b has been assumed. Figures 7(a)

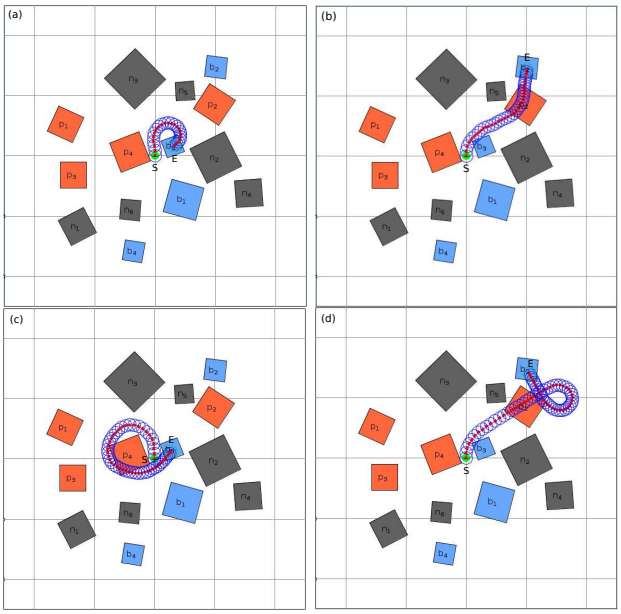

Figure 5: Paths for ψ_m, ψ_b, ψ_{s^1} and ψ_{s^2}.

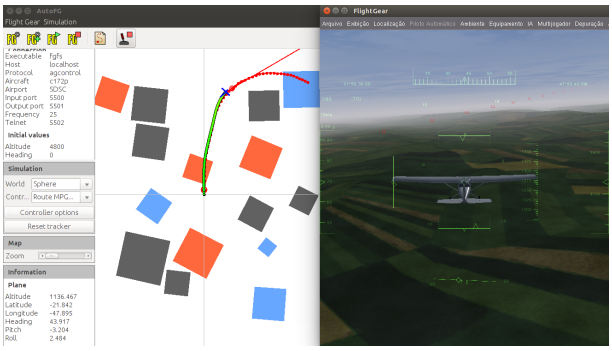

Figure 6: Framework for FlightGear with MPGA paths.

and (c) show the paths when there are winds with 10 knots during the flight simulation, while Figures 7 (b) and (d) have the same path execution under winds with 50 knots. The winds bring disturbances that do not allow to follow exactly the MPGA2 route in both maps. As expected, the difference between MPGA2 path and the path executed by autopilot is larger under strong winds, but the aircraft stays away from non-fly zones and near the MPGA2 path for both maps. Videos of these simulations are available at web[2].

VI. Conclusions

This paper presented a path re-planning problem to land a UAV under four types of critical situation. The objective was to minimize damages increasing the safety during the landing. Two strategies for planner were introduced, GH and MPGA, where it was evaluated the MPGA with GH (MPGA2) to initialize paths (individuals) and without it

[2]http://lcrserver.icmc.usp.br/projects/uav/wiki

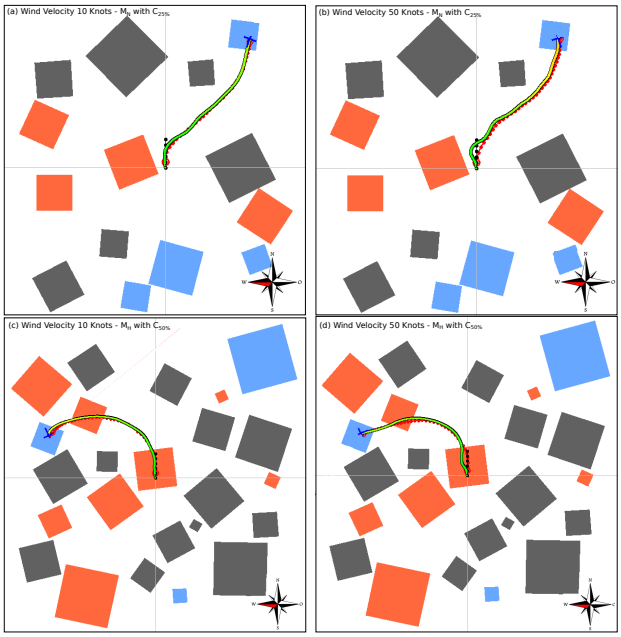

Figure 7: (a), (c) FG simulation with winds 10 knots, (b) and (d) with winds 50 knots. Wind direction: west.

(MPGA1). A set of 600 maps were generated to evaluate the proposed methods. The results showed that all approaches were able to land the aircraft for more than 67% of maps on average under all critical situations. MPGA2 was the most promising method once it took advantage to evolve individuals initialized by GH, being able to land the UAV in safety regions for more than 79% of the maps. The methods were better dealing with battery problem and worse for motor failure, so this must be taken into account as future works to improve them. MPGA approaches returned solutions around 1 sec, while GH spent 0.07 sec. Thus, GH returned fast solution, but it can fail more than MPGA1 and MPGA2 to land in bonus regions. The simulation with FlightGear showed that the aircraft was able to follow the path even under different wind velocities. Thus, the proposed methods found good quality solutions within a short computational time, which is relevant for such problem. As future work, the mathematical model will be improved aiming to describe this problem as a mixed-integer linear programming model and solve it using exact methods. The functions to describe critical situations will also be improved and other failures will be considered. Finally, a deeper evaluation of many land paths will be conducted based on the results from flight simulator as FlightGear.

ACKNOWLEDGMENTS

This paper acknowledges the support of FAPESP projects 2014/12297-0, 2014/11331-0 and 2013/26091-2.

REFERENCES

[1] E. Besada-Portas, L. Torre, A. Moreno, and J. L. Risco-Martín. On the performance comparison of multi-objective evolutionary uav path planners. *Inf. Sci.*, pages 111–125, July 2013.

[2] L. Blackmore, M. Ono, and B. C. Williams. Chance constrained optimal path planning with obstacles. IEEE Press, 2011.

[3] K. Branco, J. Pelizzoni, L. Oliveira Neris, O. Trindade, F. Osorio, and D. Wolf. Tiriba - a new approach of uav based on model driven development and multiprocessors. In *ICRA*, pages 1–4. IEEE, May 2011.

[4] R. Clarke and L. B. Moses. The regulation of civilian drones' impacts on public safety. *Computer Law & Security Review*, 30(3):263 – 285, 2014.

[5] P. M. França, A. Mendes, and P. Moscato. A memetic algorithm for the total tardiness single machine scheduling problem. *European Journal of Operational Research*, 132(1):224 – 242, 2001.

[6] H. X. Li. *Kongming: A Generative Planner for Hybrid Systems with Temporally Extended Goals*. 237 p., Massachusetts Institute of Technology, 2010.

[7] A. L. P. Mattei, E. Fonseca, N. M. Figueira, O. Trindade, and F. Vaz. Uav in-flight awareness: A tool to improve safety. In *5th European Conference for Aeronautics and Space Sciences (EUCASS)*, Munich, 2013.

[8] N. Meuleau, C. Neukom, C. Plaunt, D. E. Smith, and T. Smith. The emergency landing planner experiment. In *21st International Conference on Automated Planning and Scheduling*, 2011.

[9] N. Meuleau, C. Plaunt, D. E. Smith, and T. B. Smith. An emergency landing planner for damaged aircraft. In K. Z. Haigh and N. Rychtyckyj, editors, *IAAI*. AAAI, 2009.

[10] Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4:1–32, 1996.

[11] P. Moscato, A. Mendes, and R. Berretta. Benchmarking a memetic algorithm for ordering microarray data. *Biosystems*, 88(1-2):56–75, 2007.

[12] M. Ono, B. C. Williams, and L. Blackmore. Probabilistic planning for continuous dynamic systems under bounded risk. *J. Artif. Int. Res.*, 46(1):511–577, Jan. 2013.

[13] Y. V. Pehlivanoglu. A new vibrational genetic algorithm enhanced with a voronoi diagram for path planning of autonomous uav. *Aerospace Science and Technology*, 16(1):47 – 55, 2012.

[14] C. F. M. Toledo, M. Arantes, R. R. R. de Oliveira, and B. Almada-Lobo. Glass container production scheduling through hybrid multi-population based evolutionary algorithm. *Appl. Soft Comput.*, 13(3):1352–1364, 2013.

[15] X. Zhang and H. Duan. An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning. pages 270–284, 2015.